投资学案例分析作业(附 Python 代码实现)

金融工程　杨宸宇 2016301550186

说明：

本人使用 mac 电脑的 Excel 求解最优化问题时，发现 Excel 的 Solver 插件无法使用，实在无法用本机的 Excel 实现资产配置设计，所以采用 Python 语言进行实现(附录从第五页开始)

1.

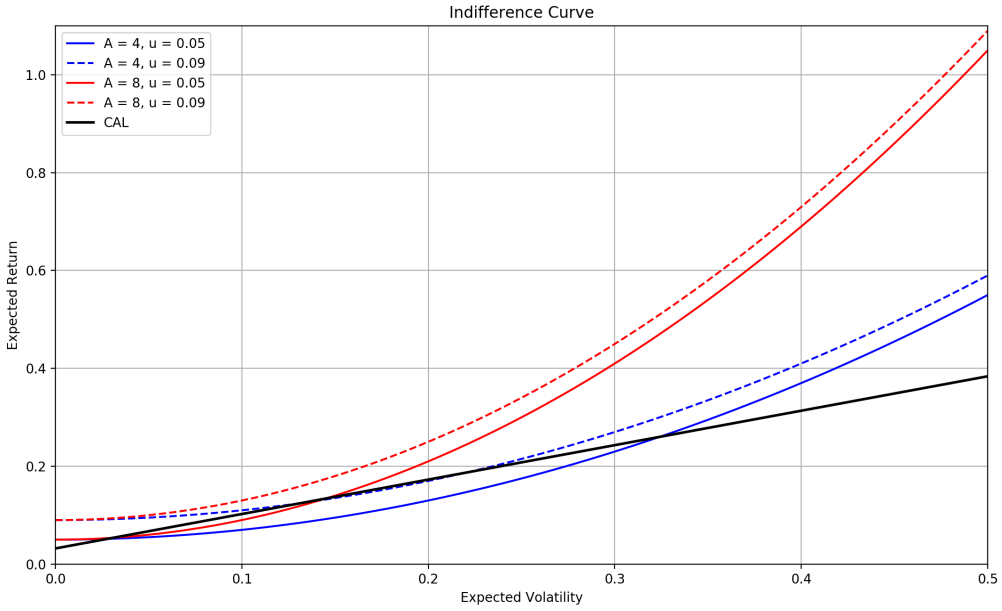
当前的资产配置并不是最优配置(图和表格见第二题)，针对不同的风险偏好的医院，我选取了 4 种情况对风险资产配置进行举例：

首先运用方法：

$$y^* = \frac{E(r_p) - r_f}{A\delta^2}$$

求出了在 A = 8 时，风险资产配比最优为 73.23%， 在 A = 16 时， 风险资产配比最优为 36.63%

接着选择了 A = 4 和 8，u = 0.05 和 0.09 的情况绘制了 CAL 和无差异曲线，如图所示：

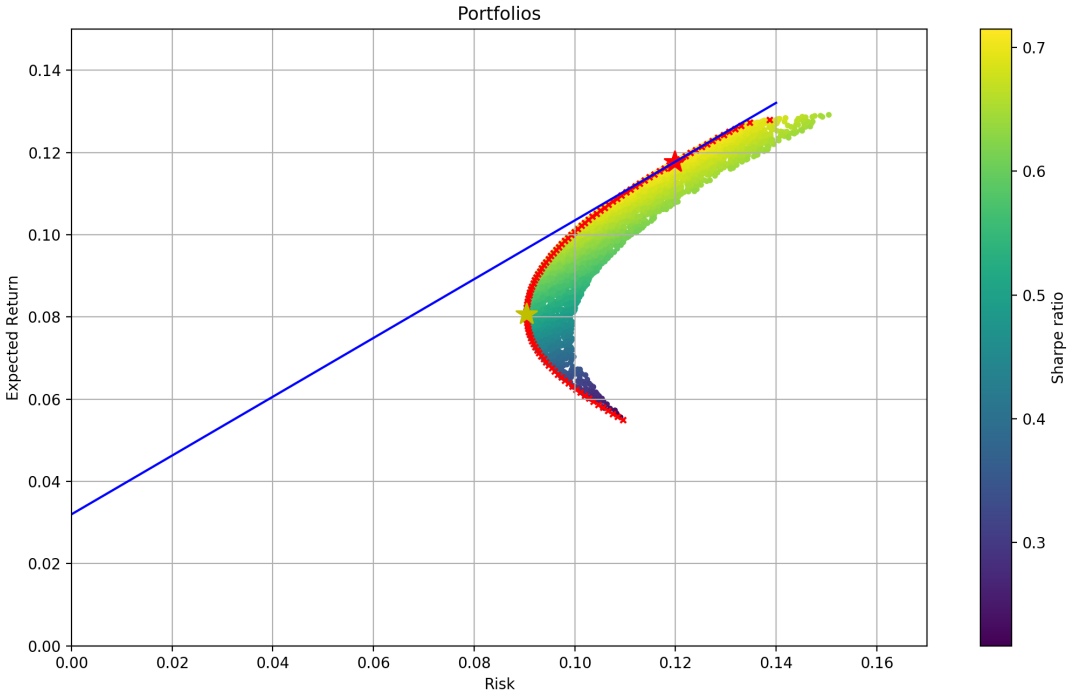


2.

本题首先的图像首先我用蒙特卡洛方法模拟了 10000 次随机取样，将每次资产的预期风险与预期收益的定画在了图上，并通过颜色深度区分 Sharpe 的大小，

接着通过凸优化求出了资产配置的有效边界(图上红色叉)，最大 Sharpe 的点(图上红色星位置)，最小方差的点(图上黄色星位置)，本题的最优组合分配比例及图像如下所示：

Portfolio	Expected Ret.	Stdev	US Equity	Foreign Equity	Bonds
Max_Sharpe	11.77%	11.99%	40.64%	47.06%	12.30%
Min_Variance	8.07%	9.04%	4.18%	33.58%	62.24%



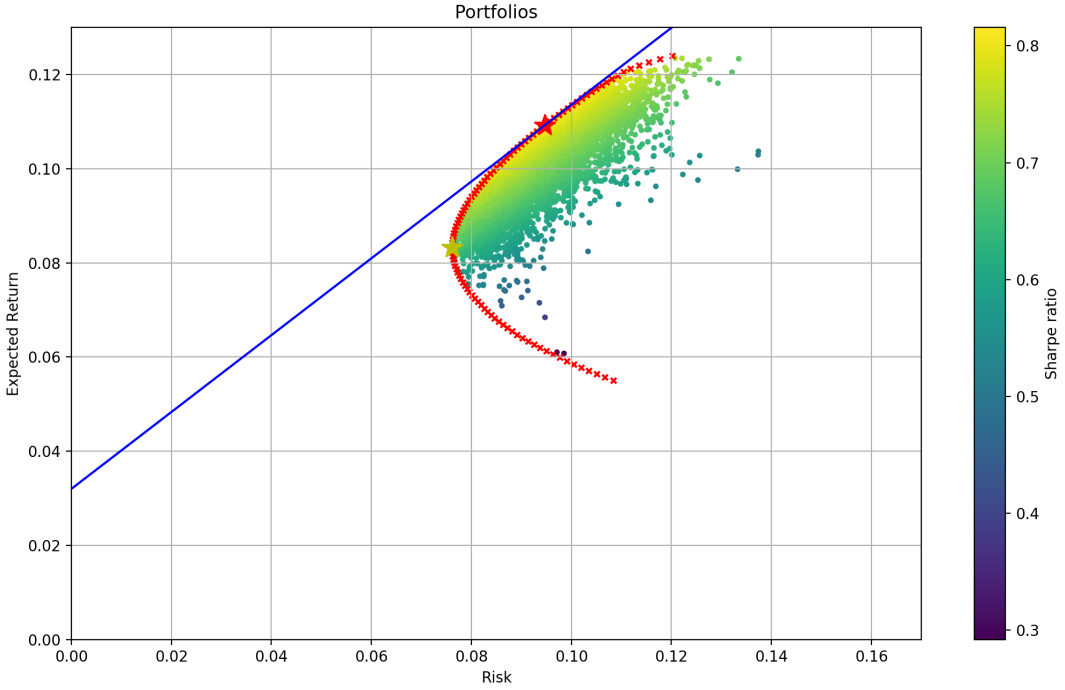
通过观察本图的最优 Sharpe 处的 Portfolio 中各个资产比例，我们可以明确的发现 LTP 的最优配置情况和报告中的配置情况不相符合，由此可见报告中并不是最优资产组合。

3.

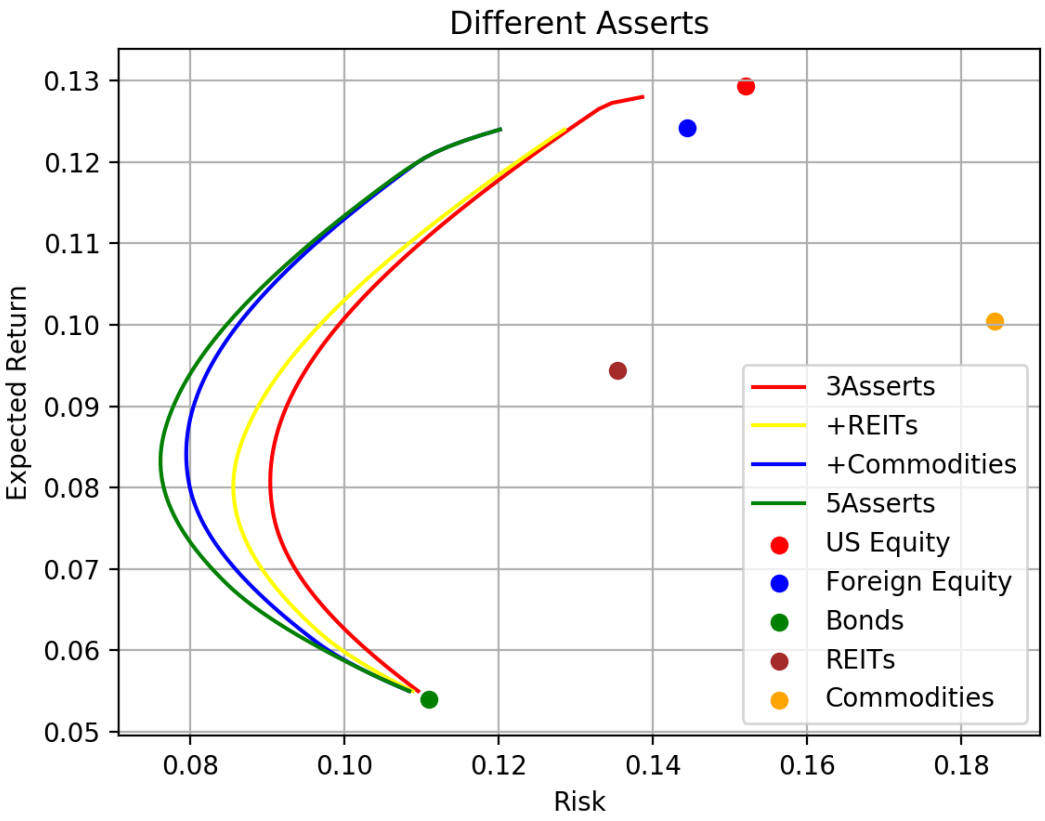
本题首先的图像首先我用蒙特卡洛方法模拟了 10000 次随机取样，将每次资产的预期风险与预期收益的定画在了图上，并通过颜色深度区分 Sharpe 的大小。

接着通过凸优化求出了资产配置的有效边界(图上红色叉)，最大 Sharpe 的点(图上红色星位置)，最小方差的点(图上黄色星位置)，本题的最优组合分配比例及图像如下所示：

Portfolio	Expected Ret.	Stdev	US Equity	Foreign Equity	Bonds	REITs	Commodities
Max_Sharpe	10.92%	9.46%	24.37%	30.84%	10.82%	0.09887556	24.09%
Min_Variance	8.33%	7.61%	6.16E-18	18.71%	44.18%	18.13%	18.97%



之后，将原始三种资产组合，添加 REITs 的组合，添加 commodities 的组合以及五种资产组合的 risk-return 曲线进行对比，画在同一幅图中，如图所示：



我们可以发现整体上新的资产配置优于原始的三种资产的配置组合，其中 Bonds 能很好的降低整体风险，其他的资产具有相对较好的预期收益，如 US Equity，也有风险相对较高的 Commodities，综合考虑还是 5 种资产的配置组合效果最佳。

附录：代码

(Python)

```
# 投资学案例分析作业
# 作者：杨宸宇
# 学号：2016301550186

# 调用所需的包
import os
import pandas as pd
import numpy as np
from sympy import Symbol
from sympy.solvers import solve
from scipy.optimize import minimize
import matplotlib.pyplot as plt



# 第一题----------------------------------------------

# 导入准备好的 excel 数据，单独赋值
os.chdir(path = r'/Users/mac/Desktop')
data = pd.read_excel('Investment.xlsx')
data.set_index(keys=data.iloc[:,0], inplace = True)

# 转换为 array 方便后面运算
expected_return = np.array(data.iloc[0:3, 1])
standard_variance = np.array(data.iloc[0:3, 2])
correlation = np.array(data.iloc[0:3, 3:6])
# 计算协方差矩阵
corvariance = standard_variance * correlation * standard_variance\
        .reshape(3, 1)
# 由题目知
weight = np.array([0.55, 0.3, 0.15])
STP_expected_return = 0.032

# 计算 LTP 的预期收益率和波动性
LTP_expected_return = np.sum(weight * expected_return)
LTP_expected_volatility = np.sqrt(np.dot(weight.T, \
    np.dot(corvariance, weight)))

# 最优组合配比
def opt(bias):
    return (LTP_expected_return - STP_expected_return) \
                / (bias  * np.square(LTP_expected_volatility))

# bias = 16
y1 = opt(16)
```

```
# bias = 8
y2 = opt(8)



# 画无差异曲线取 u = 0.05 和 u = 0.09, A = 4(bias) 和 A = 8(bias)

# A = 4, u = 0.05
A = 4
u = 0.05
standard_variances = np.linspace(0, 0.5, 100)
expected_returns = []
for std in standard_variances:
    x = Symbol('x')
    expected_return = solve(x - 0.5 * A * np.square(std) - u, x)[0]
    expected_returns.append(expected_return)
standard_variances = np.array(standard_variances)
expected_returns = np.array(expected_returns)

plt.plot(standard_variances, expected_returns, \
            label = 'A = 4, u = 0.05', c = 'blue')

# A = 4, u = 0.09
A = 4
u = 0.09
standard_variances = np.linspace(0, 0.5, 100)
expected_returns = []
for std in standard_variances:
    x = Symbol('x')
    expected_return = solve(x - 0.5 * A * np.square(std) - u, x)[0]
    expected_returns.append(expected_return)
standard_variances = np.array(standard_variances)
expected_returns = np.array(expected_returns)

plt.plot(standard_variances, expected_returns, \
            label = 'A = 4, u = 0.09', c = 'blue', linestyle = '--')

# A = 8, u = 0.05
A = 8
u = 0.05
standard_variances = np.linspace(0, 0.5, 100)
expected_returns = []
for std in standard_variances:
    x = Symbol('x')
    expected_return = solve(x - 0.5 * A * np.square(std) - u, x)[0]
    expected_returns.append(expected_return)
standard_variances = np.array(standard_variances)
expected_returns = np.array(expected_returns)

plt.plot(standard_variances, expected_returns, \
```

```
            label = 'A = 8, u = 0.05', c = 'red')


# A = 8, u = 0.09
A = 8
u = 0.09
standard_variances = np.linspace(0, 0.5, 100)
expected_returns = []
for std in standard_variances:
    x = Symbol('x')
    expected_return = solve(x - 0.5 * A * np.square(std) - u, x)[0]
    expected_returns.append(expected_return)
standard_variances = np.array(standard_variances)
expected_returns = np.array(expected_returns)

plt.plot(standard_variances, expected_returns, \
            label = 'A = 8, u = 0.09', c = 'red', linestyle = '--')

# CAL
k = (LTP_expected_return - STP_expected_return) / LTP_expected_volatility
b = 0.032
y = 0.5 * k + b
plt.plot([0, 0.5], [0.032, y], label = 'CAL', linewidth = 2,c = 'black')
plt.axis([0, 0.5, 0, 1.1])
plt.grid(True)
plt.xlabel('Expected Volatility')
plt.ylabel('Expected Return')
plt.title('Indifference Curve')
plt.legend()
plt.show()
```

```
# 第二题----------------------------------------------

# 导入准备好的 excel 数据, 单独赋值
os.chdir(path = r'/Users/mac/Desktop')
data = pd.read_excel('Investment.xlsx')
data.set_index(keys=data.iloc[:,0], inplace = True)

# 转换为 array 方便后面运算
expect_return = np.array(data.iloc[0:3, 1])
standard_variance = np.array(data.iloc[0:3, 2])
correlation = np.array(data.iloc[0:3, 3:6])

# 计算协方差矩阵
corvariance = standard_variance * correlation * standard_variance\
```

```
        .reshape(3, 1)

# 采用蒙特卡洛方法来来绘制资产组合的分布图
expect_returns = []
risks = []
for i in range(10000):
    weights = np.random.random(3)
    # 随机生成资产权重
    weights /= np.sum(weights)
    por_return = np.sum(weights * expect_return)
    por_risk = np.sqrt(np.dot(weights.T , np.dot(corvariance , weights)))
    expect_returns.append(por_return)
    risks.append(por_risk)
# matplotlib.pyplot 倾向于 np.array 格式，因此进行转换
expect_returns = np.array(expect_returns)
risks = np.array(risks)
# 无风险利率定为 3.2%(文章中 2005 年 STP 的收益率)
risk_free_rate = 0.032

plt.scatter(risks, expect_returns, \
    c = (expect_returns - risk_free_rate) / risks, marker = 'o', s = 8)
plt.grid(True)
plt.xlabel('Risk')
plt.ylabel('Expected Return')
plt.title('Portfolios')
plt.colorbar(label = 'Sharpe ratio')



# 找出 Sharpe 最大的点，利用 Scipy.optimize.minimize 进行凸优化

max_Sharpe = lambda x: - ((np.sum(x * expect_return) - risk_free_rate) /\
    np.sqrt(np.dot(x.T, np.dot(corvariance, x))))
cons = {'type':'eq', 'fun':lambda x: np.sum(x) - 1}
bnds = tuple((0,1) for x in range(3))
x0 = np.array([1/3, 1/3, 1/3])
# 将优化结果储存到 result 中
result = minimize(max_Sharpe, x0, method = 'SLSQP', bounds = bnds, \
    constraints = cons)
weight = result['x']
expected_return = np.sum(weight * expect_return)
risk = np.sqrt(np.dot(weight.T , np.dot(corvariance , weight)))
# A 点为 Sharpe 值最大的资产组合在图中的坐标
A = (risk, expected_return)
# 画出 Sharpe 最大的点
plt.plot(risk, expected_return, 'r*', markersize = 15.0)
# 为了美观确定坐标轴的刻度范围
plt.axis([0, 0.17, 0, 0.15])
```

```
# 找出 Risk 最低的点

min_risk = lambda x: np.sqrt(np.dot(x.T, np.dot(corvariance, x)))
cons = {'type':'eq', 'fun':lambda x: np.sum(x) - 1}
bnds = tuple((0,1) for x in range(3))
x0 = np.array([1/3, 1/3, 1/3])
result = minimize(min_risk, x0, method = 'SLSQP', bounds = bnds, \
    constraints = cons)
weight = result['x']
expected_return = np.sum(weight * expect_return)
risk = np.sqrt(np.dot(weight.T , np.dot(corvariance , weight)))
# 画出 Risk 最低的点
plt.plot(risk, expected_return, 'y*', markersize = 15.0)


# 寻找并画出有效边界
expected_returns = np.linspace(0.055 ,0.128, 100)
risks = []
for i in expected_returns:
    cons = ({'type':'eq', 'fun':lambda x: np.sum(x) - 1},\
        {'type':'eq', 'fun':lambda x: np.sum(x * expect_return) - i})
    result = minimize(min_risk, x0, method = 'SLSQP', \
        constraints = cons, bounds = bnds)
    risks.append(result['fun'])
plt.scatter(risks, expected_returns, c = 'red', marker = 'x', s = 14)



# 画出资产配置线，理论上应该凸优化求出最大 k，但因为之前求了 Sharpe 最大组合，因此选择直接画
k = (A[1] - 0.032) / A[0]
b = 0.032
x = 0.14
y = k * x + b
plt.plot([0, 0.14], [0.032, y], c = 'blue')


# 显示图像
plt.show()









# 第三题----------------------------------------------

# 导入准备好的 excel 数据,单独赋值,文件位置在本机的 desktop 处
os.chdir(path = r'/Users/mac/Desktop')
data = pd.read_excel('Investment.xlsx')
data.set_index(keys=data.iloc[:,0], inplace = True)
```

```
# 转换为 array 方便后面运算
expect_return = np.array(data.iloc[:, 1])
standard_variance = np.array(data.iloc[:, 2])
correlation = np.array(data.iloc[:, 3:])

# 计算协方差矩阵
corvariance = standard_variance * correlation * standard_variance\
        .reshape(5, 1)

# 采用蒙特卡洛方法来来绘制资产组合的分布图
expect_returns = []
risks = []
for i in range(10000):
    weights = np.random.random(5)
    weights /= np.sum(weights)
    por_return = np.sum(weights * expect_return)
    por_risk = np.sqrt(np.dot(weights.T , np.dot(corvariance , weights)))
    expect_returns.append(por_return)
    risks.append(por_risk)
expect_returns = np.array(expect_returns)
risks = np.array(risks)
# 无风险利率定为 3.2%(文章中 2005 年 STP 的收益率)
risk_free_rate = 0.032

plt.scatter(risks, expect_returns, \
    c = (expect_returns - risk_free_rate) / risks, marker = 'o', s = 8)
plt.grid(True)
plt.xlabel('Risk')
plt.ylabel('Expected Return')
plt.title('Portfolios')
plt.colorbar(label = 'Sharpe ratio')



# 找出 Sharpe 最大的点

max_Sharpe = lambda x: - ((np.sum(x * expect_return) - risk_free_rate) /\
    np.sqrt(np.dot(x.T, np.dot(corvariance, x))))
cons = {'type':'eq', 'fun':lambda x: np.sum(x) - 1}
bnds = tuple((0,1) for x in range(5))
x0 = np.array([0.2, 0.2, 0.2, 0.2, 0.2])
result = minimize(max_Sharpe, x0, method = 'SLSQP', bounds = bnds, \
    constraints = cons)
weight = result['x']
expected_return = np.sum(weight * expect_return)
risk = np.sqrt(np.dot(weight.T , np.dot(corvariance , weight)))
A = (risk, expected_return)
# 画出 Sharpe 最大的点
plt.plot(risk, expected_return, 'r*', markersize = 15.0)
```

```
plt.axis([0, 0.17, 0, 0.13])


# 找出Risk最低的点

min_risk = lambda x: np.sqrt(np.dot(x.T, np.dot(corvariance, x)))
cons = {'type':'eq', 'fun':lambda x: np.sum(x) - 1}
bnds = tuple((0,1) for x in range(5))
x0 = np.array([0.2, 0.2, 0.2, 0.2, 0.2])
result = minimize(min_risk, x0, method = 'SLSQP', bounds = bnds, \
    constraints = cons)
weight = result['x']
expected_return = np.sum(weight * expect_return)
risk = np.sqrt(np.dot(weight.T , np.dot(corvariance , weight)))
# 画出Risk最低的点
plt.plot(risk, expected_return, 'y*', markersize = 15.0)
# 找出并画出有效边界
expected_returns = np.linspace(0.055 ,0.124, 100)
risks = []
for i in expected_returns:
    cons = ({'type':'eq', 'fun':lambda x: np.sum(x) - 1},\
        {'type':'eq', 'fun':lambda x: np.sum(x * expect_return) - i})
    result = minimize(min_risk, x0, method = 'SLSQP', \
        constraints = cons, bounds = bnds)
    risks.append(result['fun'])
plt.scatter(risks, expected_returns, c = 'red', marker = 'x', s = 14)



# 画出资产配置线
k = (A[1] - 0.032) / A[0]
b = 0.032
x = 0.12
y = k * x + b
plt.plot([0, 0.12], [0.032, y], c = 'blue')


# 显示图像
plt.show()
```